HOWTO-Suricata IDS on Debian 6.0 (Squeeze)

Miguel Angel Cabrerizo, doncicuto[at]gmail.com

v0.3, 20 October 2011

This is a how-to for installing Suricata IDS on Debian 6.0. At the time of this writing only installation is covered. This how-to uses Stein Gjoen's template for small HOWTOs (http://www.nyx.net/~sgjoen/mintplt.html)

---

1. Introduction

On July 1, 2010 the *Open Information Security Foundation* released the first stable version of Suricata IDS. I'm a long time Snort user but I want to know more about this IDS so that´s why I wrote this how-to for Debian 5.0 (Lenny). A year later, I've updated this how-to for Debian 6.0.

Suricata is ready for using PF_RING the new network socket that, according to *ntop's web* (what a great tool ntop is...), dramatically improves the packet capture speed.

Suricata installation is not difficult but it needs a little time if you want to use PF_RING. This how-to uses the INSTALL and INSTALL.PF_RING files that comes with Suricata but with some modifications on my own.

The latest version of this document can be found at diatel.wordpress.com. I hope this document helps you in using Suricata.

1.1 Copyright

1.2 Disclaimer

Use the information in this document at your own risk. I disavow any potential liability for the contents of this document. Use of the concepts, examples, and/or other content of this document is entirely at your own risk.

All copyrights are owned by their owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements.

You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

1.3 Credits

In this version I have the pleasure of acknowledging

```
Victor Julien and William Metcalf
Stein Gjoen
```

Any comments or suggestions can be mailed to my mail address on Gmail: doncicuto [at] gmail.com

2. Installation

2.1 Installing the prerequisites

After a fresh installation of Debian 6.0 (Lenny) you will need to download the following packages and install them. Change the Linux headers package according to your server's configuration.

I will install Suricata using the Root account.

```
# apt-get install build-essential libpcre3-dev libpcap-dev libnet1-dev libyaml-
dev libnetfilter-queue-dev zlib1g-dev htp subversion flex bison linux-headers-
2.6.32-5-686 dkms libcap-ng-dev
```

2.2 Configure PF_RING

This is the toughest part and there are many steps, so try not to get lost and be patient!

```
# cd /usr/src
# svn --force export https://svn.ntop.org/svn/ntop/trunk/PF_RING/
PF_RING_CURRENT_SVN
# mkdir /usr/src/pf_ring-4
# cp -Rf /usr/src/PF_RING_CURRENT_SVN/kernel/* /usr/src/pf_ring-4/
# cd /usr/src/pf_ring-4/
```

Using your favourite editor, create a file called dkms.conf and place the following into the file.

PACKAGE_NAME="pf_ring"
PACKAGE_VERSION="4"
BUILT_MODULE_NAME[0]="pf_ring"
DEST_MODULE_LOCATION[0]="/kernel/net/pf_ring/"
AUTOINSTALL="yes"

```
# dkms add -m pf_ring -v 4
```

If the previous command ran successfully you will receive the message `"DKMS: add Completed".`

```
# dkms build -m pf_ring -v 4
```

If the previous command ran successfully you will receive the message `"DKMS: build Completed".`

```
# dkms install -m pf_ring -v 4
```

If the previous command ran successfully you will receive the message `"DKMS: install Completed".`

As we are working with SVN you may have to change the driver name, the version or the location

for the driver.

```
# mkdir /usr/src/e1000e-pf_ring-1.3.10a
# cp -Rf /usr/src/PF_RING_CURRENT_SVN/drivers/PF_RING_aware/intel/e1000e/e1000e-
1.3.10a/src/* /usr/src/e1000e-pf_ring-1.3.10a/
# cp -f /usr/src/PF_RING_CURRENT_SVN/kernel/linux/pf_ring.h /usr/src/e1000e-
pf_ring-1.3.10a/
# cd /usr/src/e1000e-pf_ring-1.3.10a/
# sed -i -e
's/\.\.\/\.\.\/\.\.\/\.\.\/\.\.\/\.\.\/kernel\/linux\/pf\_ring\.h/pf\_ring\.h/'
netdev.c
```

Again, using your favourite editor, create a file called dkms.conf and place the following into the file.

PACKAGE_NAME="e1000e-pf_ring"
PACKAGE_VERSION="1.3.10a"
BUILT_MODULE_NAME[0]="e1000e"
DEST_MODULE_LOCATION[0]="/kernel/drivers/net/e1000e/"
AUTOINSTALL="yes"

Now, let´s use dkms.

```
# dkms add -m e1000e-pf_ring -v 1.3.10a
```

If the previous command ran successfully you will receive the message "`DKMS: add Completed`". If you find an error, you may have to use the following command to remove the module "`dkms remove -m e1000e-pf_ring -v 1.3.10a –all`"

Warning. I had problems building the e1000e-pf_ring module with kernel 2.6.32-5.

```
/usr/src/e1000e-pf_ring-1.3.10a/netdev.c: In function âe1000_runtime_resumeâ:
/usr/src/e1000e-pf_ring-1.3.10a/netdev.c:6353: error: âstruct dev_pm_infoâ has
no member named âruntime_autoâ
/usr/src/e1000e-pf_ring-1.3.10a/netdev.c: At top level:
/usr/src/e1000e-pf_ring-1.3.10a/netdev.c:7184: error: implicit declaration of
function âSET_RUNTIME_PM_OPSâ
/usr/src/e1000e-pf_ring-1.3.10a/netdev.c:7186: error: initializer element is not
constant
/usr/src/e1000e-pf_ring-1.3.10a/netdev.c:7186: error: (near initialization for
âe1000_pm_ops.suspend_noirqâ)
```

I had no choice but edit netdev.c and comment the following lines.

```
#ifdef CONFIG_PM_OPS
#ifdef HAVE_SYSTEM_SLEEP_PM_OPS
/*static const struct dev_pm_ops e1000_pm_ops = {
        SET_SYSTEM_SLEEP_PM_OPS(e1000_suspend, e1000_resume)
        SET_RUNTIME_PM_OPS(e1000_runtime_suspend,
                           e1000_runtime_resume, e1000_idle)
};*/
#endif /* HAVE_SYSTEM_SLEEP_PM_OPS */
#endif

static int e1000_runtime_resume(struct device *dev)
{
        struct pci_dev *pdev = to_pci_dev(dev);
        struct net_device *netdev = pci_get_drvdata(pdev);
        struct e1000_adapter *adapter = netdev_priv(netdev);
```

```
        if (!e1000e_pm_ready(adapter))
                return 0;

/*      adapter->idle_check = !dev->power.runtime_auto; */
        return __e1000_resume(pdev);
}


#ifdef HAVE_SYSTEM_SLEEP_PM_OPS
/*      .driver.pm = &e1000_pm_ops,  */
#else
        .suspend  = e1000_suspend,
        .resume   = e1000_resume,
#endif /* HAVE_SYSTEM_SLEEP_PM_OPS */
```

Phew!

Ok, now let's try to build it!

```
# dkms build -m e1000e-pf_ring -v 1.3.10a
# dkms install -m e1000e-pf_ring -v 1.3.10a
```

If the previous command ran successfully you will receive the message "`DKMS: install Completed`".

Ok, we are finishing, don´t panic! Final steps:

```
# mkdir -p /opt/PF_RING/{bin,lib,include/linux,sbin}
# cp -f /usr/src/PF_RING_CURRENT_SVN/kernel/linux/pf_ring.h
/opt/PF_RING/include/linux/
# cd /usr/src/PF_RING_CURRENT_SVN/userland/lib
# sed -i -e 's/\@INSTALL_PREFIX\@/\/opt\/PF_RING/' Makefile.in
# cp -f pfring_mod_dna.h /opt/PF_RING/include
#./configure
# make &&  make install

# cd /usr/src/PF_RING_CURRENT_SVN/userland/libpcap-1.1.1-ring
# sed -i -e 's/\.\.\/lib\/libpfring\.a/\/opt\/PF_RING\/lib\/libpfring\.a/'
Makefile.in
# ./configure --prefix=/opt/PF_RING && make && make install

# cd /usr/src/PF_RING_CURRENT_SVN/userland/tcpdump-4.1.1
# sed -i -e 's/\.\.\/lib\/libpfring\.a/\/opt\/PF_RING\/lib\/libpfring\.a/'
Makefile.in
# sed -i -e 's/-I \.\.\/libpcap-1\.0\.0-ring/-I \/opt\/PF_RING\/include/'
Makefile.in
# sed -i -e 's/-L \.\.\/libpcap-1\.0\.0-ring\/-L /\/opt\/PF_RING\/lib\//'
Makefile.in
# ./configure LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib"
--prefix=/opt/PF_RING/ --enable-ipv6 && make && make install
```

2.3 Configuring and installing Suricata

I'm going to configure Suricata with the following features:

- --enable-pfring (better packet capture performance)
- --with_libpcap-libraries (libpcap ready for PFRING)
- --with_libcap_ng-libraries (for dropping privileges)
- --enable-nfqueue (IPS capabilities)

- --with-libhtp-libraries (HTP HTML pre-processor)

First ,download Suricata current release to a directory, in my case, /opt.
At the time of writing, Suricata stable version is 1.0.5.

Using the prefix option, suricata will be installed in /opt/PF_RING

```
# cd /opt
# wget http://www.openinfosecfoundation.org/download/suricata-1.0.5.tar.gz
# tar xvfz suricata-1.0.5.tar.gz
# cd suricata-1.0.5
#  ./configure --enable-pfring --with-libpfring-libraries=/opt/PF_RING/lib
--with-libpfring-includes=/opt/PF_RING/include --with-libpcap-
libraries=/opt/PF_RING/lib --with-libpcap-includes=/opt/PF_RING/include
LD_RUN_PATH="/opt/PF_RING/lib:/usr/lib:/usr/local/lib" --prefix=/opt/PF_RING/
--enable-nfqueue --with-libcap_ng-libraries=/usr/lib --with-libhtp-libraries
# make
# make install
```

You're ready to configure Suricata.

3. Basic configuration

3.1 Suricata user

It is always advisable to create a user account, with no privileges, to run Suricata.

```
# groupadd suricata
# useradd -g suricata suricata -s /sbin/nologin
```

3.2 Directories

Create a directory to store your logs and give permissions to the suricata user.

```
# mkdir /var/log/suricata
# chown suricata:suricata /var/log/suricata
```

Now you will need to create a directory to store configuration files

```
# mkdir /etc/suricata
```

Finally create a directory to store the rules

```
# mkdir /etc/suricata/rules
```

3.3 Config files

Now it's time to move Suricata's config files to its directory

```
# cp /opt/suricata-1.0.5/suricata.yaml /etc/suricata/
# cp /opt/suricata-1.0.5/classification.config /etc/suricata/
```

It's always good to have a threshold.conf file in case you want to disable rules or set a threshold for alerts.

```
# touch /etc/suricata/threshold.config
```

Spend a few minutes reviewing the well-documented suricata.yaml file.

These are my suggestions:

- As we are not using Prelude in this how-to comment the lines for alert-prelude

```
# - alert-prelude:
#    enabled: no
#    profile: suricata
```

- Change the HOME_NET variable.
- Add your servers to the HTTP_SERVERS, SMTP_SERVERS....
- I find interesting that Suricata logs to a file

  outputs:

```
- console:
     enabled: yes

- file:
     enabled: yes
     filename: /var/log/suricata.log
```

3.4 Rules

There's no fun if your IDS/IPS has no rules so let's get some.

Emerging Threats is an open source community project managed by Matt Jonkman which provides free Snort and Suricata signatures.

Download the latest rules and move them to the rules directory. ET rules will offer you the highest level of compatibility with Suricata.

```
# cd /usr/src
# wget http://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz
# tar xvfz emerging.rules.tar.gz
# mv /usr/src/rules/*.rules /etc/suricata/rules/
```

3.5 Change ownership

Don´t forget to change the ownership

```
# chown -R suricata:suricata /etc/suricata/
```

3.6 Let's run Suricata for the first time!

```
# /opt/PF_RING/bin/suricata -c /etc/suricata/suricata.yaml -i eth0
```

And... this is what we have

```
 -- all 5 packet processing threads, 3 management threads initialized, engine
started.
```

Perfect.

New chapters about configuration and log analysis will be added soon.

Please *contact me* if you have found any *errors* or any *questions*